Generation of Executable State-Based Models for Efficient Performance Evaluation of Embedded System Architectures

Sébastien Le Nours

University of Nantes, France

February 6, 2014
Embedded system architecture

- An **architecture** is an organization of components that is elaborated under constraints.
- It designates also the principles guiding its design.
Purpose of the system architecting process

- Creation of components’ organization and definition of components’ characteristics.
- **Performance evaluation**: analysis of resource usage over time.
Architecture design flow

System requirements

What?

System specifications

How?

System architecting

Specification of hardware

High level co-simulation

Specification of software

Hardware description languages

Low level co-simulation

Software programming languages

Gates

Detailed co-simulation

Assembly code

Real prototype for validation and verification
Time description accuracy

- Different levels of accuracy are used during the architecture design process.
- Efficiency of models is a matter of compromise between accuracy, simulation speed, and modelling effort.
Time description accuracy

- Different levels of accuracy are used during the architecture design process.
- Efficiency of models is a matter of compromise between accuracy, simulation speed, and modelling effort.

- **Real prototype**: Execution close to real-time, but very long design cycle.

![Diagram showing the process of system architecting and usage of resources]

- **Usage of resources**: Measurements
  - P3
  - P2
  - N
  - P1

- **Execution time**: $t_s$
Different levels of accuracy are used during the architecture design process.

Efficiency of models is a matter of compromise between accuracy, simulation speed, and modelling effort.

Continuous time model: Very accurate execution but very slow, long design cycle.
Different levels of accuracy are used during the architecture design process.

Efficiency of models is a matter of compromise between accuracy, simulation speed, and modelling effort.

**Discrete time model**: Accurate execution but with limited speed, long design cycle.
Time description accuracy

- Different levels of accuracy are used during the architecture design process.
- Efficiency of models is a matter of compromise between accuracy, simulation speed, and modelling effort.

**Discrete event model**: Fast execution with limited accuracy, reduced design cycle.
Current trends in the embedded system domain

Observation

- Dissemination, interaction, heterogeneity ...
- Design complexity is related to integration of more and more heterogeneous components
- Needs of methods to facilitate the system architecting process and to allow performance evaluation

Presented contribution

An approach (methods, models, and tool) to facilitate the creation of efficient performance models with light modelling effort.

S. Le Nours (Univ. Nantes, FR)
Current trends in the embedded system domain

Observation

- Dissemination, interaction, heterogeneity ...
- Design complexity is related to integration of more and more heterogeneous components
- Needs of methods to facilitate the system architecting process and to allow performance evaluation

Presented contribution

An approach (methods, models, and tool) to facilitate the creation of efficient performance models with light modelling effort.
Presentation schedule

1. Context of the study
2. Addressed topic
3. Proposal and results
4. Opened prospects
5. Sum-up and conclusion
Principles of performance evaluation approaches

System specifications

Functional architecture model

Platform resources model

Mapping

Physical architecture model

Generation

Executable model

- Discrete event simulation
- Performance evaluation according to working scenarios
- Tuning of architecture parameters
## Current performance evaluation approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Functional architecture model</th>
<th>Framework</th>
<th>Platform model accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCE [1]</td>
<td>PSM</td>
<td>SCE</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Daedalus [2]</td>
<td>KPN</td>
<td>Sesame</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Metropolis [4]</td>
<td>PN</td>
<td>_</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Arpinen et al. [5]</td>
<td>UML/KPN</td>
<td>_</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Kreku et al. [6]</td>
<td>UML</td>
<td>VTT_ABSOLUT</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Bourgos et al. [7]</td>
<td>KPN</td>
<td>BIP</td>
<td>Task accurate</td>
</tr>
<tr>
<td>SpaceCoDesign [8]</td>
<td>PN</td>
<td>SpaceStudio</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Expermeta [9]</td>
<td>SysML</td>
<td>MetaSyn</td>
<td>Task accurate</td>
</tr>
<tr>
<td>Mirabilis [10]</td>
<td>PN</td>
<td>VisualSim</td>
<td>Task accurate</td>
</tr>
</tbody>
</table>
Illustration: architecture performance modelling

- Functional architecture modelling: ideal organization of performed communication and computation.

- **Workload model**: expression of communication and computation loads with no complete functionality description required.

```
F1 F3
F2
F0 M1 F1 M2 M3 M4 F3 M5 F4
M6
F1 while(1) {
  read(M1,token);
  execute(token);
  write(M2,token);
  execute(token);
  write(M3,token);
}
F3 while(1) {
  read(M2,token);
  execute(token);
  read(M4,token);
  execute(token);
  write(M5,token);
}
F2 while(1) {
  read(M3,token);
  execute(token);
  write(M4,token);
}
F4 while(1) {
  read(M5,token);
  execute(token);
  write(M6,token);
}
```
Physical architecture modelling: organization of communication and computation performed under constraints.

Platform model: kind of processing resources, scheduling and arbitration policies, ...
Trace-driven simulation

- The platform resources are driven by **traces** generated by the application execution.
- The **mapping layer** manages the concurrency offered by the modeled platform.
- The **simulation kernel** manages events and the advancement of the simulation time.
Trace-driven simulation

- The platform resources are driven by traces generated by the application execution.
- The **mapping layer** manages the concurrency offered by the modeled platform.
- The **simulation kernel** manages events and the advancement of the simulation time.

```
while(1) {
    read(M1,token);
    execute(token);
    write(M2,token);
    execute(token);
    write(M3,token);
}
```

![Diagram](image.png)

F1: \(T_{i1}(0)\)

Resource usage

\(x_{M_1}(0)\)

\(t_s\) simulation time

S. Le Nours (Univ. Nantes, FR)
Trace-driven simulation

- The platform resources are driven by **traces** generated by the application execution.
- The **mapping layer** manages the concurrency offered by the modeled platform.
- The **simulation kernel** manages events and the advancement of the simulation time.
Trace-driven simulation

- The platform resources are driven by **traces** generated by the application execution.
- The **mapping layer** manages the concurrency offered by the modeled platform.
- The **simulation kernel** manages events and the advancement of the simulation time.

```
while(1) {
    read(M1,token);
    execute(token);
    write(M2,token);
    execute(token);
    write(M3,token);
}
```

**Generation of executable state-based models**

February 6, 2014 12 / 39
• Possible observation of platform resource usage, power consumption, memory size, ...

• Simulation of concurrent processes over the simulation time implies calls to the simulation kernel and time consuming context switches.

Illustration: architecture performance model simulation

Resource usage

Calls to the simulation kernel

Simulation time $t_s$

F1: /execute()

F2: /execute()

F3: /execute() /execute()

F4: /execute()

$P_1$ $P_2$

$x_{M_1}(0)$ $x_{M_2}(0)$ $x_{M_3}(0)$ $x_{M_4}(0)$ $x_{M_5}(0)$ $x_{M_6}(0)$

$T_{i_1}(0)$ $T_{i_2}(0)$ $T_{j_3}(0)$ $T_{i_4}(0)$ $T_{i_1}(1)$

S. Le Nours (Univ. Nantes, FR)
The compromise between simulation speed and accuracy limits the achievable complexity of architecture models.

How to limit the number of required calls to the simulation kernel?
Related work

- Combination of different levels of accuracy and adaptation during simulation
  - *Multi-Accuracy Power and Performance Transaction-Level Modeling* [12]
  - *Accuracy-adaptive simulation of transaction level models* [13]

- Retroactive correction methods
  - *Result-oriented modeling - a novel technique for fast and accurate TLM* [14]

- Combination of simulation and formal methods
  - *Combining simulation and formal methods for system-level performance analysis* [16]
  - *Analytical timing estimation for temporally decoupled TLMs considering resource conflicts* [17]
Main points

- **A new approach** for performance model generation, offering good compromise between simulation speed and accuracy.
- **A computation method** of architecture model evolution instants to limit the number of calls to the simulation kernel.
- The **state-based model** concept: combination between simulation and algebraic models.
- A **generator tool** for automatic creation of executable state-based models of architectures.
Computation method of model evolution instants

- Reduction of the number of calls to the simulator by dynamic computation of model evolution instants.

![Diagram showing the computation method of model evolution instants](image-url)
Reduction of the number of calls to the simulator by dynamic computation of model evolution instants.

Equivalent model

\[ M_1 \xrightarrow{F_1} M_2 \xrightarrow{F_2} M_3 \xrightarrow{F_3} M_4 \]

Simulation kernel

\[ \text{Calls to the kernel} \]

\[ x_{M_1}(k) \quad x_{M_2}(k) \quad x_{M_3}(k) \quad x_{M_4}(k) \]

\[ x_{M_1}(k+1) \quad x_{M_2}(k+1) \quad x_{M_3}(k+1) \]

S. Le Nours (Univ. Nantes, FR)
Computation method of model evolution instants

- Reduction of the number of calls to the simulator by dynamic computation of model evolution instants.

\[ M_1 M_2 M_3 M_4 \]

\[ F_1 \quad F_2 \quad F_3 \]

\[ T_1(k) \quad T_2(k) \quad T_3(k) \quad T_4(k) \]

\[ x_{M_1}(k) \quad x_{M_2}(k) \quad x_{M_3}(k) \quad x_{M_4}(k) \]

\[ x_{M_1}(k+1) \quad x_{M_2}(k+1) \quad x_{M_3}(k+1) \quad x_{M_4}(k+1) \]

\[ \text{Equivalent model} \]

\[ x_{M_2}(k) = \max(x_{M_3}(k-1), T_1(k)+x_{M_1}(k)) \]

\[ x_{M_3}(k) = \max(x_{M_4}(k-1), T_2(k)+x_{M_2}(k)) \]

\[ x_{M_4}(k) = T_3(k)+x_{M_3}(k) \]

\[ \text{Calls to the kernel} \]

\[ \text{Simulation kernel} \]

\[ \text{Mapping} \]

- S. Le Nours (Univ. Nantes, FR)
Temporal decoupling method

- Preservation of the model accuracy by introduction of an observation time that evolves locally.

![Diagram](image-url)

Resource usage

- \(x_{M1}(k)\), \(x_{M2}(k)\), \(x_{M3}(k)\), \(x_{M4}(k)\), \(x_{M4}(k+1)\)

- \(T_1(k+1)\), \(T_2(k+1)\), \(T_3(k+1)\)

- \(N\)

- Simulation kernel

- Mapping
Temporal decoupling method

- Preservation of the model accuracy by introduction of an observation time that evolves locally.

### Equivalent model

- \( x_{M2}(k) = \max(x_{M3}(k-1), T_1(k) + x_{M1}(k)) \)
- \( x_{M3}(k) = \max(x_{M4}(k-1), T_2(k) + x_{M2}(k)) \)
- \( x_{M4}(k) = T_3(k) + x_{M3}(k) \)
General observations

- Parts of an architecture model can be replaced by an equivalent model that computes, during simulation, specific instants when resource usage evolves.
- A reduction by a factor of N leads to a simulation speed-up in the same order of magnitude.
- Computation depends on the characteristics of the given architecture model.
- The complexity of the computation method depends on the number of elements that are replaced.
General observations

- Parts of an architecture model can be replaced by an equivalent model that computes, during simulation, specific instants when resource usage evolves.
- A reduction by a factor of N leads to a simulation speed-up in the same order of magnitude.
- Computation depends on the characteristics of the given architecture model.
- The complexity of the computation method depends on the number of elements that are replaced.
General observations

- Parts of an architecture model can be replaced by an equivalent model that computes, during simulation, specific instants when resource usage evolves.
- A reduction by a factor of N leads to a simulation speed-up in the same order of magnitude.
- Computation depends on the characteristics of the given architecture model.
- The complexity of the computation method depends on the number of elements that are replaced.
General observations

- Parts of an architecture model can be replaced by an equivalent model that computes, during simulation, specific instants when resource usage evolves.
- A reduction by a factor of N leads to a simulation speed-up in the same order of magnitude.
- Computation depends on the characteristics of the given architecture model.
- The complexity of the computation method depends on the number of elements that are replaced.
More about discrete event systems

- Systems which evolution is based on two main properties: synchronization and concurrency.
- Evolution captured using two formalisms: timed automata or Petri nets.

![Diagram of a discrete event system with transitions and states labeled with variables and events.]

$$x_{M1}(0) \quad x_{M2}(0) \quad x_{M3}(0) \quad x_{M4}(0) \quad x_{M1}(1) \quad x_{M5}(0) \quad x_{M2}(1) \quad x_{M6}(0)$$

T_{i1}(0) \quad T_{j1}(0) \quad T_{i2}(0) \quad T_{j3}(0) \quad T_{i4}(0) \quad T_{i1}(1) \quad T_{i3}(0) \quad T_{j3}(0) \quad T_{i4}(0) \quad T_{i1}(1)$$

T_s
Algebraic description of discrete event systems

- Convenient expression of evolution instants using the (max, plus) algebra [18].
- Algebraic framework based on two operators: maximization, $\oplus$, and addition, $\otimes$.

$$\begin{cases} 
  x_{M2}(k) = \max(T_1(k) + x_{M1}(k), x_{M3}(k - 1)) \\
  x_{M3}(k) = T_2(k) + x_{M2}(k)
\end{cases}$$

becomes

$$\begin{cases} 
  x_{M2}(k) = T_1(k) \otimes x_{M1}(k) \oplus x_{M3}(k - 1) \\
  x_{M3}(k) = T_2(k) \otimes x_{M2}(k)
\end{cases}$$

- Allows linear and non linear state equations to be formed:

$$\begin{cases} 
  X(k) = f(X, U) \\
  Y(k) = g(X, U)
\end{cases}$$
Set of equations:

\[
\begin{align*}
X(k) &= A(k, 0)X(k) \oplus A(k, 1)X(k - 1) \\
Y(k) &= C(k, 0)X(k)
\end{align*}
\]

Achieved simulation speed-up: \( \times 2.33 \), with preserved accuracy.
Set of equations:

\[
\begin{aligned}
X(k) &= A(k, 0)X(k) \oplus A(k, 1)X(k - 1) \\
Y(k) &= C(k, 0)X(k)
\end{aligned}
\]

Achieved simulation speed-up: \(\times 2.33\), with preserved accuracy.
Extension to the general case

- Definition of **executable state-based models**.
- Based on a set of equations related to the evolution instants of architecture elements.
- Use of an executable model that expresses dependencies with external elements.

\[
X(k) = f(X, U) \\
Y(k) = g(X, U)
\]

Simulation time \( t_s \)

\[
u_1(k) u_1(k+1) u_{Nu}(k)
\]

\[
y_1(k) y_{Ny}(k) y_1(k+1)
\]
Evaluation method

- Comparison between architecture models simulated using the trace-driven approach and equivalent state-based models.
- Comparison of accuracy and simulation time.
- Analysis of different kinds of architectures with different numbers of simulation events saved.
Evaluation of the approach

- Evaluation of achieved simulation speed-up for different ratios of saved events.

Simulation speed-up

<table>
<thead>
<tr>
<th>Ratio of event saved (%)</th>
<th>Simulation speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(11)</td>
</tr>
<tr>
<td>2</td>
<td>(20)</td>
</tr>
<tr>
<td>4</td>
<td>(29)</td>
</tr>
<tr>
<td>6</td>
<td>(38)</td>
</tr>
<tr>
<td>8</td>
<td>(47)</td>
</tr>
</tbody>
</table>

Ratio of event saved (%) 0 10 20 30 40 50 60 70 80 90 100
Evaluation of the approach

- Evaluation of the computation method complexity on achieved simulation speed-up.

![Graph showing evaluation of simulation speed-up with varying X sizes and number of instants.](image-url)
Evaluation of the computation method complexity on achieved simulation speed-up.
The compromise between simulation speed and accuracy can be improved in many significant cases using the proposed approach.

The best simulation speed-up achieved is a factor of 30 with no loss of accuracy.

The presented approach is applicable for different kind of architecture models.

Possibility to automate the creation process of state equations.
Implementation of the state equations

- Introduction of **temporal dependency graphs** for implementation of equations.
- In case of non linear set of equations, additional control statements are required.
- The complexity of the computation method is related to the size of the graph.
Generator tool

Selected elements of the architecture description

Application relation analysis
Identification of evolution instants

Definition of graph nodes (i)

Application behaviour analysis
Identification of temporal dependencies

Definition of graph arcs (ii)

Platform concurrency analysis
Identification of evolution instants and temporal dependencies

Definition of graph nodes and arcs (iii)

Mapping analysis
Identification of durations

Definition of graph arcs' weights (iv)

Temporal dependency graph description

Model to text transformation process

S. Le Nours (Univ. Nantes, FR)
Case study

- Receiver part of the LTE physical layer.
- Evaluation of the processing resources’ usage over time.
- Evaluation of 3 different mappings.
Case study

- Achieved simulation speed-up by a factor of 4 with preserved accuracy.
- The related temporal dependency graph is made of 11 nodes.
- Non linear set of equations is implemented using the proposed approach.
Presentation schedule

1. Context of the study
2. Addressed topic
3. Proposal and results
4. Opened prospects
5. Sum-up and conclusion
Extension of the proposed approach

- Influence of dynamic scheduling and arbitration policies.
- Influence of possible preemption of communication and processing resources.
The proposed approach leads to the definition of a **generic execution model**.

Significant reduction of the modelling effort can be considered by using this execution model.
About the needs of methodology...
Refinement approach of performance models

- About the needs of methodology...
About the needs of methodology...
About the needs of methodology...
Presentation sum-up

- Presentation of **an approach to improve the compromise between simulation speed and accuracy** in performance models of embedded system architectures.
- Originality of the approach is related to the **combination between simulation and formal models**.
- Proposal of **a new automatic generation approach** of efficient executable models.
- It allows performance models of architectures to be automatically abstracted and more complex architectures to be addressed.
References


